Newton Aitken

Juan Angarita, Héctor Rodríguez, Aldemar Ramirez

9/8/2019

## Problema

Hallar la raíz de una función a partir de un a traves del método de Newton en combinación con el método de aceleración de Aitken

## Solución

**Lenguaje de programación:**  R

## Función 1: Método de Newton

Parametros:

fun <- función

x0 <- desde donde se comienza la busqueda de la raíz

tol <- tolerancia mínima que debe tener la función

maxiter <- cantidad máxima de iteraciones

Valores de retorno:

x1 <- resultado de la raíz

errorAbsoluto <- vector de errores absolutos de las

errorRelativo <- vector de errores relativos de las

x <- vector de las

vectorx <- vector en donde se almacena los valores para la función Aitken.

## Implementación

secuencia = function(fun, x0, tol, maxiter){  
  
 numiter = 0  
 errorAbsoluto = c()  
 errorRelativo = c()  
 x = c()  
 vectorx=c()  
 g = parse(text=fun) # parse devuelve tipo "expression"  
 g. = D(g,"x")  
 fx = function(x){eval(g)} # convertir f a funciÃ³n  
 fp = function(x){eval(g.)} # convertir fâa funciÃ³n  
 correccion = -fx(x0)/fp(x0)  
  
 while (abs(correccion) >= tol && numiter <= maxiter) {  
 numiter = numiter + 1  
 if ((fp(x0) == 0)) stop("División por cero")  
   
 x1 = x0 + correccion  
 x0 = x1  
  
 vectorx <- c(vectorx, x1) #Vector a usar para aitken  
 if(numiter==4){ #Devuelve el vector  
 return(vectorx)  
 }  
 x <- c(x,x1)  
 errorAbsoluto <- c(errorAbsoluto,abs(correccion))  
 errorRelativo <- c(errorRelativo,abs(correccion)/(abs(x0)))  
  
 correccion = -fx(x1)/fp(x1)  
  
 }  
 if (numiter > maxiter){ warning("Se alcanzó el máximo número de iteraciones.")  
 return(vectorx)  
 } else {  
 return(vectorx)  
 }  
 }  
  
##------------------------------------------------------------------------------  
  
## SE REALIZA LA COPIA DEL METODO NEWTON RAPHSON CON EL OBJETIVO DE REALIZAR LA COMPARACION   
  
newtonraphson = function(fun, x0, tol, maxiter){  
 numiter = 0  
 errorAbsoluto = c()  
 errorRelativo = c()  
 x = c()  
 vectorx=c()  
 g = parse(text=fun) # parse devuelve tipo "expression"  
 g. = D(g,"x")  
 fx = function(x){eval(g)} # convertir f a funciÃ³n  
 fp = function(x){eval(g.)} # convertir fâa funciÃ³n  
 #cat("FUNCION: ", fun ,"\n")  
 #cat("DERIVADA: ", fp(1) ,"\n")  
 correccion = -fx(x0)/fp(x0)  
   
 while (abs(correccion) >= tol && numiter <= maxiter) {  
 numiter = numiter + 1  
 if ((fp(x0) == 0)) stop("División por cero")  
 x1 = x0 + correccion  
 x0 = x1  
   
 x <- c(x,x1)  
 errorAbsoluto <- c(errorAbsoluto,abs(correccion))  
 errorRelativo <- c(errorRelativo,abs(correccion)/(abs(x0)))  
   
 correccion = -fx(x1)/fp(x1)  
   
 }  
 if (numiter > maxiter){ warning("Se alcanzó el máximo número de iteraciones.")  
 my\_list <- list("resultado" = x1, "errorAbsoluto" = errorAbsoluto, "errorRelativo" = errorRelativo, "x" = x)  
 return(my\_list)  
 } else {  
   
 my\_list <- list("resultado" = x1, "errorAbsoluto" = errorAbsoluto, "errorRelativo" = errorRelativo, "x" = x)  
 return(my\_list)  
 }  
}

En esta ocasion la diferencia principal entre la función llamada secuencia y newtonraphson es que la primera únicamente devuelve el vector con la secuencia de valores necesarios para realizar el método de Aitkens.

La función de secuencia básicamente obtiene los valores de las primeras tres iteraciones luego de aplicar el método de Newtonraphson. La razón de que se hagan sólo tres iteraciones es que dicha es la cantidad de valores necesarios en una secuencia para aplicar el método de aitken.

## Función 2: Método de Aitken

Parametros:

vector <- arreglo con 3 valores [ , , ]

Valor de retorno:

valor <- valor de un nuevo tras aplicar el método de Aitken.

aitken=function(vectors){  
 valor=0;  
   
 x0=vectors[1]  
 x1=vectors[2]  
 x2=vectors[3]  
   
 valor=x2 -(((x2-x1)^2)/(x2-x1)-(x1-x0));  
 cat("El valor de aitken es:",valor,"\n")  
 return(valor);  
}

La secuencia es recibida por el método de aitken que luego de operar los contenidos del vector para acelerar la convergencia retorna un valor que será usado en la función de Newtonraphson de manera normal.

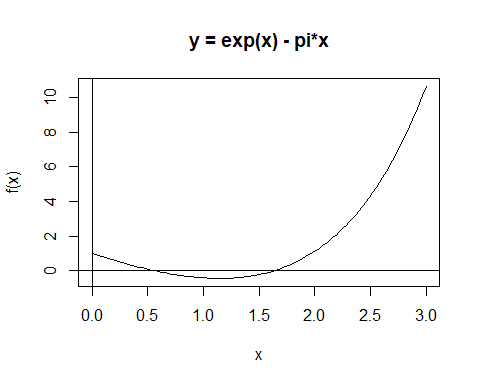
De esta manera, una aproximacion corregida de en la funcion newtonraphson se realiza como

Mientras que en la método de aitken se obtienen los tres primeros valores y se obtiene un nuevo para una nueva secuencia.

**Resultados**

**Gráfica de la función**

f = function(x) exp(x) - pi\*x  
curve(f, 0,3); abline(h=0, v=0) #gráfico para decidir un intervalo  
title(main="y = exp(x) - pi\*x")



Se puede apreciar que las raices de la función se encuentra en el intervalo [1.5,2], por esta razón una buena aproximación inicial para un seria

Se muestra una tabla con la cantidad de iteraciones y el resultado de cada una para la función . Se realizan separaciones para diferenciar la implementación de la función newton\_Aitken de la newton\_Raphson.

## --- Pruebas  
# recibe la función como una tira  
## Secuencia   
resultados<-newtonraphson("exp(x)-pi\*x",1.5, 1e-8, 100)  
secuencia3<-secuencia("exp(x)-pi\*x",1.5, 1e-8, 100)  
tablaErrores <- data.frame(  
 "iteraciones" = 1:length(resultados$errorAbsoluto),  
 "x\_n" = resultados$x,  
 "errorAbsoluto" = resultados$errorAbsoluto,  
 "errorRelactivo" = resultados$errorRelativo  
)  
  
print(tablaErrores)

## iteraciones x\_n errorAbsoluto errorRelactivo  
## 1 1 1.672152 1.721517e-01 1.029522e-01  
## 2 2 1.639892 3.225952e-02 1.967173e-02  
## 3 3 1.638531 1.361402e-03 8.308674e-04  
## 4 4 1.638528 2.380175e-06 1.452630e-06

cat("Método de Aitken: \n")

## Método de Aitken:

valornuevo<-aitken(secuencia3)

## El valor de aitken es: 1.607633

cat("Se usa el valor ",valornuevo," para aplicar newton nuevamente \n")

## Se usa el valor 1.607633 para aplicar newton nuevamente

resultados2<-newtonraphson("exp(x)-pi\*x",valornuevo, 1e-8, 100)  
  
tablaErrores <- data.frame(  
 "iteraciones" = 1:length(resultados2$errorAbsoluto),  
 "x\_n" = resultados2$x,  
 "errorAbsoluto" = resultados2$errorAbsoluto,  
 "errorRelativo" = resultados2$errorRelativo  
)  
print(tablaErrores)

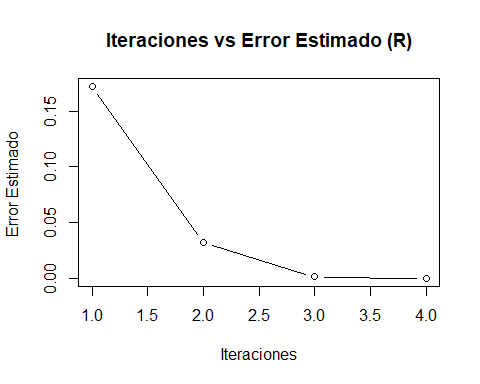
## iteraciones x\_n errorAbsoluto errorRelativo  
## 1 1 1.639830 3.219714e-02 1.963444e-02  
## 2 2 1.638531 1.299227e-03 7.929221e-04  
## 3 3 1.638528 2.167647e-06 1.322923e-06

En un primer análisis observamos como la función en la que se usó el método de aitken requirió de un menor número de iteraciones para llegar a la raiz.

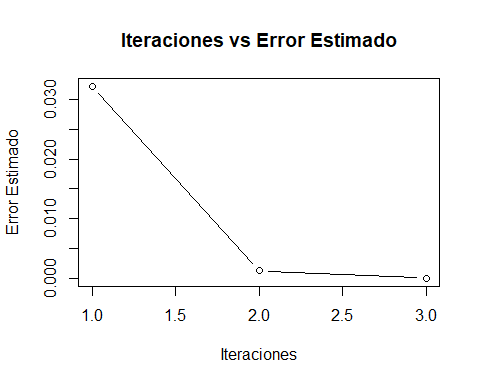
**Grafica de iteraciones vs error estimado**

Las gráficas que representan la implementación de la función newton\_raphson se identifican con una (R) en el titulo.

plot(x = 1:length(resultados$x), y = resultados$errorAbsoluto,  
 xlab = "Iteraciones", ylab = "Error Estimado ",   
 type="b", main = "Iteraciones vs Error Estimado (R)")



##--------------------------------------------------------------------------------------------  
plot(x = 1:length(resultados2$x), y = resultados2$errorAbsoluto,   
 xlab = "Iteraciones", ylab = "Error Estimado",   
 type="b", main = "Iteraciones vs Error Estimado")

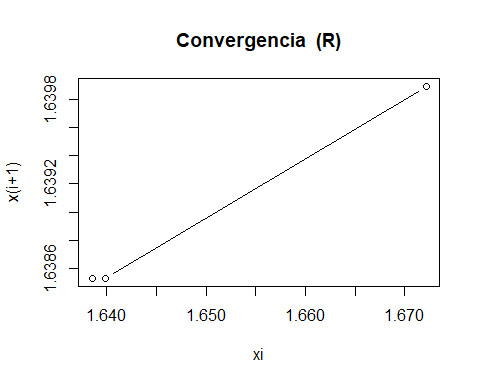


El error absoluto estimado en ambos casos converge a 0 y para el caso de aitken tanto la magnitud del error estimado como la cantidad de iteraciones son menores a las del newtonraphson normal.

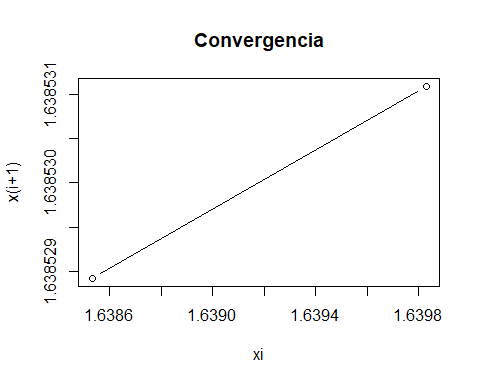
para la función newtonraphson y método de aitken

**Grafica de x(i) vs x(i+1)**

m\_i = resultados$x[-length(resultados$x)]  
m\_i2 = resultados$x  
m\_i2 = m\_i2[-1]  
  
plot(x =m\_i, y =m\_i2, xlab = "xi", ylab = "x(i+1)", type="b",main = "Convergencia (R)")



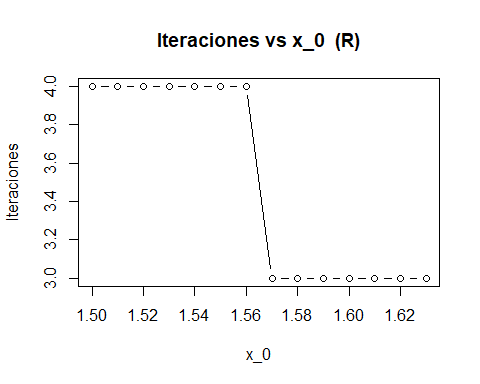
##--------------------------------------------------------------------------------------------  
m\_i = resultados2$x[-length(resultados2$x)]  
m\_i2 = resultados2$x  
m\_i2 = m\_i2[-1]  
  
plot(x =m\_i, y =m\_i2, xlab = "xi", ylab = "x(i+1)", type="b",main = "Convergencia")



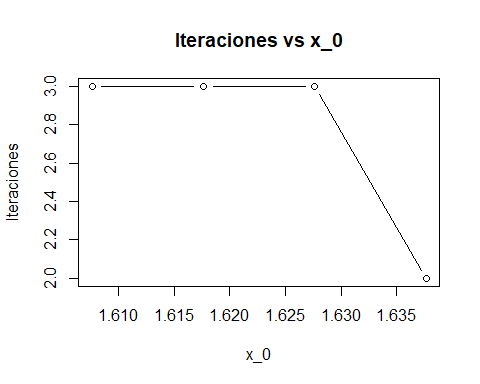
De acuerdo con las gráficas se establece que ambos métodos tienen una convergencia cuadrática.

**Grafica de iteraciones vs x0**

ls=1.5  
vs = c()   
iteracioness = c()  
while(ls<1.64) {  
 vs <- c(vs,ls)  
 resultados<-newtonraphson("exp(x)-pi\*x",ls, 1e-8, 100)  
   
 tablaErrores <- data.frame(  
 "iteraciones" = 1:length(resultados$errorAbsoluto),  
 "x\_n" = resultados$x,  
 "errorAbsoluto" = resultados$errorAbsoluto,  
 "errorRelactivo" = resultados$errorRelativo  
 )  
 iteracioness <- c(iteracioness,length(resultados$errorAbsoluto))  
 ls = ls+0.01  
 #print(tablaErrores)  
}  
plot(x =vs, y =iteracioness, xlab = "x\_0", ylab = "Iteraciones", type="b",main = "Iteraciones vs x\_0 (R)")



##--------------------------------------------------------------------------------------------  
  
l=valornuevo  
v = c()   
iteraciones = c()  
while(l<1.64) {  
 v <- c(v,l)  
 resultados2<-newtonraphson("exp(x)-pi\*x",l, 1e-8, 100)  
   
 tablaErrores <- data.frame(  
 "iteraciones" = 1:length(resultados2$errorAbsoluto),  
 "x\_n" = resultados2$x,  
 "errorAbsoluto" = resultados2$errorAbsoluto,  
 "errorRelactivo" = resultados2$errorRelativo  
 )  
 iteraciones <- c(iteraciones,length(resultados2$errorAbsoluto))  
 l = l+0.01  
 #print(tablaErrores)  
}  
plot(x =v, y =iteraciones, xlab = "x\_0", ylab = "Iteraciones", type="b",main = "Iteraciones vs x\_0")



Mientras mas alejado se encuentre del son necesarias mas cantidad de iteraciones.

La implementacion de la función de aitken a comparacion de la newtonraphson es más eficiente en cuanto a la escogencia el .En este caso vemos como al comparar el por “defecto” de newtonraphson con el de aitken, este último requiere de un número inferior de iteraciones para encontrar la raíz, efectivamente acelerando la convergencia.